*Computers Math. Applic.* Vol. 20, No. 12, pp. 25–33, 1990
Printed in Great Britain. 

0097-4943/90 $3.00 + 0.00


# AN IMPROVED LANCZOS ALGORITHM FOR SOLVING ILL-CONDITIONED LINEAR EQUATIONS

P.-L. SHAO
North Vehicle Research Institute, P.O. Box 969-58, Beijing, People's Republic of China

(*Received* 17 *August* 1989; *in revised form* 19 *November* 1989)

**Abstract**—In the existing Lanczos algorithms for solving systems of linear equations, the estimate for the residual is effective for well-conditioned systems. However, in actual implementation on a computer we find that the estimate is no longer reliable for ill-conditioned cases. We first analyze in theory this observation, then develop an improved Lanczos algorithm. Numerical examples are also given to demonstrate the effectiveness of the present algorithm.

## 1. INTRODUCTION

Since the early 1970s, there has been considerable interest in the Lanczos algorithm for finding eigenvalues and for solving linear systems of equations [1–10]. Here we consider the latter. Up to now, several variants based on the simple Lanczos algorithm [1] have been proposed for dealing with the equation,

$$A\mathbf{x} = \mathbf{b} \tag{1.1}$$

where $A$ is a symmetric, nonsingular $n$ by $n$ matrix, and $\mathbf{b}$ is an $n$-vector.

The algorithm SYMMLQ [5] was developed by Paige and Saunders to solve indefinite systems in a stable manner. It was derived from the basic Lanczos method. Although the SYMMLQ is somewhat slower than the conjugate gradient method, it can do the work of both definite and indefinite systems. However, if the number of iterations is large the round-off errors must cause the Lanczos vectors to become linearly dependent. For this reason, Parlett proposed the Lanczos algorithm with selective orthogonalization (LANSO for short) [6–8] for solving eigenproblems and linear equations. This algorithm maintains orthogonality among the Lanczos vectors in an efficient way.

In 1984, Simon published the Lanczos algorithm with partial reorthogonalization (LANPRO for short) [9]. This reorthogonalization procedure is easier to implement than the selective orthogonalization. Both LANPRO and LANSO behave equally well for definite and indefinite systems. Unlike the SYMMLQ or the conjugate gradients, the Lanczos algorithms with re-orthogonalization can guarantee to find a solution in at most $n$ steps, and have advantages when applied to the case with several right-hand sides [8, 9]. Experiments show that the LANPRO is a better way of maintaining orthogonality for the purpose of equation solving [11]. Therefore we employ the partial reorthogonalization in this paper.

It is important to note that all the above Lanczos algorithms for linear equations and their applications use the same formula for evaluating the residual norm. However, in practical implementation, we find that the formula is reliable only for well-conditioned systems, and when applied to ill-conditioned cases it does not hold any more. We will explain theoretically the above observations in detail in Section 3, and propose a modified Lanczos algorithm to handle effectively ill-conditioned linear equations in Section 4. Finally, numerical results are given to demonstrate the reliability of the present algorithm. In this paper, we will follow the Householder convention and denote column vectors by small roman letters, matrices by capital roman letters, and scalars by small Greek letters. The symbol $\| \ \|$ denotes the 2-norm of a vector or matrix.

## 2. THE LANCZOS ALGORITHM

The basic Lanczos algorithm for solving linear equations can be simply stated as follows. The starting vector is chosen as

$$\mathbf{v}_1 = \mathbf{b}/\beta_1, \quad \beta_1 = \|\mathbf{b}\|. \tag{2.1}$$

Setting $\mathbf{v}_0 = \mathbf{0}$, for $j = 1, 2, \ldots,$

$$\alpha_j = \mathbf{v}_j^{\mathrm{T}} A \mathbf{v}_j$$

$$\beta_{j+1}\mathbf{v}_{j+1} = A\mathbf{v}_j - \alpha_j \mathbf{v}_j - \beta_j \mathbf{v}_{j-1}, \tag{2.2}$$

with $\beta_{j+1} \geqslant 0$ chosen so that $\|\mathbf{v}_{j+1}\| = 1$. After $k$ steps, we have

$$AV_k - V_k T_k = \beta_{k+1}\mathbf{v}_{k+1}\mathbf{e}_k^{\mathrm{T}}, \tag{2.3}$$

where

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \cdot & \\ & \cdot & \cdot & \beta_k \\ & & \beta_k & \alpha_k \end{bmatrix},$$

$V_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$, and $\mathbf{e}_k$ is the $k$th column of the $k$ by $k$ identity matrix $I_k$. The Lanczos vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ satisfy the relation

$$V_k^{\mathrm{T}} V_k = I_k. \tag{2.4}$$

Then we have

$$T_k = V_k^{\mathrm{T}} A V_k, \quad V_k^{\mathrm{T}}\mathbf{b} = \beta_1 \mathbf{e}_1. \tag{2.5}$$

At the $k$th step, the approximate solution to equation (1.1) can be constructed as

$$\mathbf{x}_k = V_k \mathbf{y}_k, \tag{2.6}$$

where $\mathbf{y}_k$ is obtained from the weak form of equation (1.1), i.e.

$$T_k \mathbf{y}_k = \beta_1 \mathbf{e}_1. \tag{2.7}$$

The next item is the termination criterion in which the residual size is monitored to decide when the Lanczos iteration is stopped. That is,

$$\|\mathbf{r}_k\| = \|A\mathbf{x}_k - \mathbf{b}\| = \beta_{k+1}|\phi_k|, \tag{2.8}$$

where $\phi_k$ is the $k$th element of $\mathbf{y}_k$. For later reference, it is now derived as below [8, 11, 13].

Using (2.7) and (2.3),

$$\begin{aligned} \mathbf{r}_k &= A\mathbf{x}_k - \mathbf{b} = AV_k\mathbf{y}_k - V_k\beta_1\mathbf{e}_1 \\ &= AV_k\mathbf{y}_k - V_kT_k\mathbf{y}_k = \beta_{k+1}\mathbf{v}_{k+1}\mathbf{e}_k^{\mathrm{T}}\mathbf{y}_k \\ &= \beta_{k+1}\phi_k\mathbf{v}_{k+1}. \end{aligned} \tag{2.9}$$

Hence

$$\|\mathbf{r}_k\| = \beta_{k+1}|\phi_k|.$$

In fact it is not even necessary to compute $\mathbf{y}_k$ to find $\phi_k$. There are several ways of updating $\phi_k$ from information at the previous step. A simple method is using a $QR$ factorization of $T_k$. The cost is negligible (see Ref. [11] for details).

It is worth noting that all the Lanczos algorithms mentioned above (LANPRO, LANSO and SYMMLQ) and their corresponding applications to other problems [11, 12] employ formula (2.8) as an estimate for the residual norm. In the following, we will re-examine the validity of formula (2.8).

## 3. ANALYSIS OF THE LANCZOS ALGORITHM WITH A BEING ILL-CONDITIONED

### *3.1. Two examples*

From the previous description of the existing Lanczos procedures we know that the Lanczos algorithm with partial reorthogonalization (LANPRO) is most suitable for solution of linear equations. So, we will typically examine the behavior of the LANPRO. Let us first look at the case when $A$ is well-conditioned.

*Example 3.1.* $A$ is a 60 by 60 diagonal matrix, $A = \text{diag}(1, 2, 3, \ldots, 60)$. The right-hand side $\mathbf{b} = (1, 1, \ldots, 1)^T$. In single precision computation (the round-off unit is $2^{-24}$; the machine used is Micro VAX-II), the required accuracy is that the relative residual size $\|\mathbf{r}_k\|/\|\mathbf{b}\| \leqslant 10^{-4}$. The LANPRO was terminated at step 29. The estimated residual norm by formula (2.8) and the true norm agree very well at every iteration step. The final relative residuals are $0.6683 \times 10^{-4}$ and $0.6689 \times 10^{-4}$, respectively.

Here we used a diagonal matrix for test purposes, since a diagonal matrix makes it easier to know whether the matrix is well-conditioned or not. For the nondiagonal matrix, see Example 6.3. According to Simon's argument [9, p. 124], the behavior of the algorithm for diagonal matrices is similar to that for nondiagonal matrices.

*Example 3.2.* $A$ is again a 60 by 60 diagonal matrix, $A = \text{diag}(10^{-4}, 2, 3, \ldots, 60)$, and $\mathbf{b} = (1, 1, \ldots, 1)^T$. In single precision, this problem can be considered to be ill-conditioned. To observe the whole characteristic of LANPRO, the algorithm was terminated when the maximum number of iteration steps is reached, i.e. $k = n = 60$ at which in theory the residual is zero. The estimated and the true residual, each step are shown in Fig. 1. At the end of the Lanczos run, the predicted relative residual by formula (2.8) is $0.2011 \times 10^{-22}$, whereas the true value is $0.3633 \times 10^{-2}$. We can see from Fig. 1 that the predicted curve by formula (2.8) departs with the true one to a large degree, particularly in the late stage. Consequently the Lanczos iteration might stop before the desired solution was found.

The same observation can be made for systems of higher order, e.g. $n = 1000$ (see Example 6.1). Moreover, in double precision computation we also have the same situation if the system is poorly-conditioned in this computation environment.

Next we analyze in theory the above phenomena. From now on, all the quantities refer to the computed values.

### *3.2. Theoretical analysis*

Let us examine formula (2.8) in finite precision arithmetic. The derivation of formula (2.8) in Section 2 employed relations (2.3) and (2.7). In practice, they do not hold exactly.

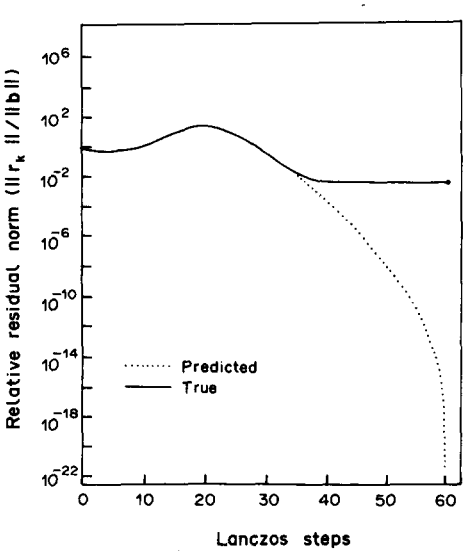


Fig. 1. Numerical behavior with $A$ being ill-conditioned.

First note that after a reorthogonalization, the current Lanczos vector becomes

$$\mathbf{v}_{j+1} - \left(\mathbf{v}'_{j+1} - \sum_{i \in L(j)} (\mathbf{v}'^{\mathrm{T}}_{j+1}\mathbf{v}_i)\mathbf{v}_i\right)\Big/\sigma, \tag{3.1}$$

where $\mathbf{v}'_{j+1}$ denotes the current Lanczos vector before reorthogonalization, $L(j)$ is determined from the partial reorthogonalization principle, and

$$\sigma = \| \mathbf{v}'_{j+1} - \sum_{i \in L(j)} (\mathbf{v}'^{\mathrm{T}}_{j+1}\mathbf{v}_i)\mathbf{v}_i \|.$$

So the three-term recurrence equation (2.2) becomes

$$\beta_{j+1}\mathbf{v}_{j+1} + \beta_{j+1}\Delta\mathbf{v}_{j+1} = A\mathbf{v}_j - \alpha_j\mathbf{v}_j - \beta_j\mathbf{v}_{j-1} - \mathbf{f}_j, \tag{3.2}$$

where $\beta_{j+1} = \sigma\beta'_{j+1}$ ($\beta'_{j+1}$ denotes $\beta_{j+1}$ before reorthogonalization),

$$\Delta\mathbf{v}_{j+1} = \frac{1}{\sigma}\sum_{i \in L(j)} (\mathbf{v}'^{\mathrm{T}}_{j+1}\mathbf{v}_i)\mathbf{v}_i \tag{3.3}$$

and $\mathbf{f}_j$ accounts for the local round-off error which satisfies

$$\| \mathbf{f}_j \| = O(\epsilon \| A \|) \tag{3.4}$$

where $\epsilon$ is the computer precision (i.e. the round-off unit). Accordingly, after $k$ steps equations (2.3) becomes

$$AV_k - V_kT_k = \beta_{k+1}\mathbf{v}_{k+1}\mathbf{e}_k^{\mathrm{T}} + \Delta V_{k+1} + F_k, \tag{3.5}$$

where $\Delta V_{k+1}$ is the $n$ by $k$ modification matrix due to reorthogonalizations. Some columns of $\Delta V_{k+1}$ consist of $\beta_{j+1}\Delta\mathbf{v}_{j+1}$ with different $j$ and the other columns are all zero vectors, i.e.

$$\Delta V_{k+1} = [\mathbf{0}, \ldots, \beta_{j+1}\Delta\mathbf{v}_{j+1}, \ldots], \quad (1 \leqslant j \leqslant k). \tag{3.6}$$

$F_k$ in equation (3.5) is the local error matrix, and its columns consist of $\mathbf{f}_j$. From equation (3.4) $F_k$ remains tiny, and can be neglected.

Because the algorithm LANPRO maintains semiorthogonality among the Lanczos vectors, the following relation [9, p. 132] holds:

$$|\mathbf{v}'^{\mathrm{T}}_{j+1}\mathbf{v}_i| \approx \sqrt{\epsilon}, \quad (i \leqslant j). \tag{3.7}$$

Therefore $\sigma \approx 1$, and we have

$$\| \Delta\mathbf{v}_{j+1} \| = O(|L(j)|\sqrt{\epsilon}), \quad (|L(j)| \leqslant j). \tag{3.8}$$

Besides, $\mathbf{v}_{j+1}$ satisfies $\| \mathbf{v}_{j+1} \| = 1$. So in relation (3.2), $\beta_{j+1}\Delta\mathbf{v}_{j+1}$ is negligible with respect to $\beta_{j+1}\mathbf{v}_{j+1}$. Since equation (3.5) is the matrix form of equation (3.2), the influence of $\Delta V_{k+1}$ on equation (3.5) can also be neglected. In summary, $\Delta V_{k+1}$ and $F_k$ have little effects on formula (2.8).

Finally, we note that the solution error of the tridiagonal system (2.7) will be large if $T_k$ is ill-conditioned. Now we express system (2.7) as

$$T_k(\mathbf{y}_k + \Delta\mathbf{y}_k) = \beta_1\mathbf{e}_1. \tag{3.9}$$

Omitting $\Delta V_{k+1}$ and $F_k$ in equation (3.5), the residual norm associated with the approximate solution $\mathbf{x}_k$ has the form:

$$\begin{aligned} \| \mathbf{r}_k \| &= \| A\mathbf{x}_k - \mathbf{b} \| = \| AV_k\mathbf{y}_k - \beta_1 V_k\mathbf{e}_1 \| \\ &= \| AV_k\mathbf{y}_k - V_kT_k(\mathbf{y}_k + \Delta\mathbf{y}_k) \| \\ &= \| (AV_k - V_kT_k)\mathbf{y}_k - V_kT_k\Delta\mathbf{y}_k \| \\ &= \| \beta_{k+1}\phi_k\mathbf{v}_{k+1} - V_kT_k\Delta\mathbf{y}_k \|. \end{aligned} \tag{3.10}$$

Now let us discuss further the influence of $\Delta\mathbf{y}_k$ on $\| \mathbf{r}_k \|$. Assume that the eigenvalues of $T_k$ and $A$ are respectively

$$\begin{aligned} &\theta_1 \leqslant \theta_2 \leqslant \cdots \leqslant \theta_k \quad (k \leqslant n), \\ &\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n. \end{aligned} \tag{3.11}$$

For simplicity, we drop the $k$ on which $\theta_i (1 \leqslant i \leqslant k)$ depends.

According to the theorem in Ref. [9], the eigenvalues of $T_k$ are (up to round-off) Rayleigh–Ritz approximations to the eigenvalues of $A$ if the Lanczos vectors are kept semiorthogonal, i.e.

$$\max_{1 \leqslant i \leqslant k} |\mathbf{v}_i^T \mathbf{v}_{k+1}| \leqslant \sqrt{\epsilon}. \tag{3.12}$$

Therefore when $k = n$, we have (up to round-off)

$$\theta_i = \lambda_i \quad (1 \leqslant i \leqslant n).$$

So, the spectral condition number of $A$ is equal to that of $T_n$, i.e.

$$\text{cond}(A) = \text{cond}(T_n). \tag{3.13}$$

For the case when $A$ is indefinite, equation (3.13) is still satisfied.

We can now conclude that if $A$ is ill-conditioned, $T_n$ must be ill-conditioned, and so may be $T_k (k < n)$ for some $k$ (see Section 4 for more details). In such cases the solution error of equation (2.7) would be great. The term $\Delta \mathbf{y}_k$ $(k \leqslant n)$ in equation (3.9) cannot be ignored and may have a strong effect on the residual $\| \mathbf{r}_k \|$. The above analysis is confirmed by Examples 3.1 and 3.2. When $A$ is well-conditioned, formula (2.8) gives a good estimate, where the effects of $\Delta V_{k+1}$, $F_k$ and $\Delta \mathbf{y}_k$ are negligible. In contrast formula (2.8) fails to predict the true residual with $A$ ill-conditioned, because of the large solution error of the tridiagonal equations.

## 4. AN IMPROVED ALGORITHM FOR ILL-CONDITIONED SYSTEMS

In order to overcome the above problem, we have to improve the algorithm LANPRO. A straightforward proposal is to compute directly $\| A \mathbf{x}_k - \mathbf{b} \|$ every step, where $\mathbf{x}_k$ is first formed. But this is considered too expensive for large matrices.

We have known that the cause of the failure of formula (2.8) is the solution error of equation (2.7). Of course, for the system which is not too ill-conditioned, the tridiagonal system can be solved in double precision to reduce round-off errors. However for very poorly-conditioned cases the whole Lanczos process should be computed in double precision. So the solution error of equation (2.7) still dominates other errors, and cannot be ignored. The situation is the same as in single precision arithmetic.

### *4.1. Making use of orthogonality*

From Example 3.2, we find that even when $A$ is ill-conditioned there still exist many steps at which the estimate of the residual size by formula (2.8) is quite accurate. This is because span$(V_k)$ has not been close enough to span$(A)$ yet, and $T_k$ is still well-conditioned for many steps, in particular at earlier steps, although $A$ is poorly conditioned. Let us make the following detailed analysis.

Consider the relation between the eigenvalues of $A$ and the eigenvalues of $T_k$. The Kaniel–Paige theory [6] shows that it is the extreme (leftmost and rightmost) eigenvalues of $A$ which are most likely to be approximated by some of the eigenvlaues of $T_k$. Namely, $(\lambda_1, \lambda_n)$, $(\lambda_2, \lambda_{n-1}), \ldots$ are most likely to be approximated by some of $\theta$s, say, $(\theta_{i1}, \theta_{i2})$, $(\theta_{i3}, \theta_{i4}), \ldots,$

Note that

$$\text{cond}(A) = \frac{\lambda_n}{\lambda_1}, \quad \text{cond}(T_k) = \frac{\theta_k}{\theta_1} = \max_{1 \leqslant l, l \leqslant k} \left( \frac{\theta_i}{\theta_l} \right). \tag{4.1}$$

When $(\theta_{i1}, \theta_{i2})$ begin to converge to $(\lambda_1, \lambda_n)$ at the $m$th step, the ratio $\theta_{i2}/\theta_{i1}$ is close to the ratio $\lambda_n/\lambda_1$. So, if $A$ is ill-conditioned $T_m$ must also be ill-conditioned at step $m$, since $\text{cond}(T_m) \geqslant \theta_{i2}/\theta_{i1}$.

For the subsequent steps, with $\theta_{i1}$ and $\theta_{i2}$ converging better to $\lambda_1$ and $\lambda_n$, the conditions of $T_{m+1}, T_{m+2}, \ldots$ will become worse than $T_m$. Thus the solution error of equation (2.7) will be larger and larger.

Therefore from the $m$th step on, we need re-predict the residual norm instead of using formula (2.8). In addition, we note that it is not sure for $\theta_{i1}$ and $\theta_{i2}$ to converge simultaneously. Here we take a conservative attitude to ensure a reliable prediction of the true residual size. On the other hand, there may exist some $T_r$ $(r < m)$ which is poorly-conditioned before the $m$th Lanczos step is reached. This is true especially in the case of indefinite $A$. However, at this step $\mathbf{x}_r$ is not close

Table 1. True and predicted residual for Example 4.1

| Iteration step | Orthogonality level | Predicted $\|\mathbf{r}_k\|$ by formula (2.8) | True residual |
|---|---|---|---|
| 1 | — | 0.20131E + 1 | 0.20131E + 1 |
| 2 | 0.6082E − 7 | 0.27968E + 1 | 0.27968E + 1 |
| 3 | 0.4682E − 6 | 0.66296E + 1 | 0.66296E + 1 |
| 4 | 0.9100E − 6 | 0.19178E + 2 | 0.19178E + 2 |
| 5 | 0.1994E − 5 | 0.53062E + 2 | 0.53062E + 2 |
| 6 | 0.4440E − 5 | 0.61821E + 2 | 0.61821E + 2 |
| 7 | 0.1851E − 4 | 0.19079E + 2 | 0.19079E + 2 |
| 8 | 0.7555E − 4 | 0.34669E + 1 | 0.34669E + 1 |
| 9 | 0.4301E − 3 | 0.12302E + 1 | 0.12298E + 1 |
| 10 | 0.4143E − 4 | 0.17929E − 5 | 0.26366E − 1 |

to the required solution, because the eigenvalues of $T_r$ do not approximate those of $A$ at all in such a case. So at this step, it is not necessary to estimate accurately $\|A\mathbf{x}_r - \mathbf{b}\|$.

The next question is how to check when the $m$th step is reached. Fortunately we can make use of Paige's result, that is, loss of orthogonality among the Lanczos vectors is equivalent to the convergence of at least one eigenpair of $T_k$ (see Ref. [6, p. 223] for a detailed description). From the previous conclusion, the converged eigenvalue (s) will be generally the extreme eigenvalue(s) of $A$. So when the orthogonality of the Lanczos vectors is gradually lost, $T_k$ will gradually become ill-conditioned. In the algorithm LANPRO, the level of orthogonality is measured by $\omega_{\max} = \max_{1 \leqslant i \leqslant k} |\omega_{ik+1}|$, where $\omega_{ik+1}$, obtained from a simple recursion, monitors the inner product $\mathbf{v}_i^{\mathrm{T}} \mathbf{v}_{k+1}$ (see Ref. [9. p. 121]). When the orthogonality level reaches $\sqrt{\epsilon}$, we find that the solution error $\Delta \mathbf{y}_k$ in formula (3.10) is usually not negligible.

Therefore when the orthogonality level $\geqslant \sqrt{\epsilon}$, the $m$th step is reached. It is interesting to note that the beginnings of the failure of formula (2.8) and the reorthogonalization in LANPRO occur at the same time. Here is a small example.

*Example 4.1.* The order of $A$ is 10, and $A = \mathrm{diag}(10^{-4}, 10^{-3}, 3, 4, \ldots, 10)$. The right-hand side $\mathbf{b} = (1, 1, \ldots, 1)^{\mathrm{T}}$. The round-off unit $\epsilon = 5.960 \times 10^{-8}$ and $\sqrt{\epsilon} = 2.441 \times 10^{-4}$. The results are presented in Table 1. We should note that at the 9th step partial reorthogonalization began to be performed, because the orthogonality level exceeded $\sqrt{\epsilon}$. At the same step, the predicted residual began to disagree with the true one.

In the above discussion, we have assumed that $A$ is positive definite. When $A$ is indefinite or negative definite, the extreme eigenvalues $\lambda_1$ and $\lambda_n$ refer to $\min(|\lambda_i|)$ and $\max(|\lambda_i|)$, where $1 \leqslant i \leqslant n$; $\theta_1$ and $\theta_k$ refer to $\min(|\theta_i|)$ and $\max(|\theta_i|)$, where $1 \leqslant i \leqslant k$. So we have the same conclusions as in the case when $A$ is positive definite.

### *4.2. Re-estimation of the residual*

We have known that the failure of formula (2.8) begins only after the $m$th step mentioned above. If the direct computation of $\|A\mathbf{x}_k - \mathbf{b}\|$ is performed at each later step, the cost would be large when $A$ is of high order. However, this problem can be removed by the following method.

In fact, $A\mathbf{x}_k$ can be accumulated efficiently as $k$ increases. The method described here is similar to the procedure used in the SYMMLQ for accumulating $\mathbf{x}_k$ (see Ref. [5, p. 621]). We first consider the stable orthogonal factorization,

$$T_k = \bar{L}_k Q_k, \quad Q_k^{\mathrm{T}} Q_k = I_k, \tag{4.2}$$

where

$$\bar{L}_k = \begin{bmatrix} \gamma_1 & & & & \\ \delta_2 & \gamma_2 & & & \\ \varepsilon_3 & \delta_3 & \gamma_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \varepsilon_k & \delta_k & \bar{\gamma}_k \end{bmatrix}$$

and $Q_k^{\mathrm{T}}$ is the product of a series of orthogonal matrices $Q_{12}, Q_{23}, \ldots, Q_{k-1k}$.

Note that

$$A\mathbf{x}_k = AV_k\mathbf{y}_k = [A\mathbf{v}_1, A\mathbf{v}_2, \ldots, A\mathbf{v}_k]\mathbf{y}_k. \tag{4.3}$$

From formula (2.2), $A\mathbf{v}_k$ is directly available when the $(k+1)$st Lanczos vector is formed. We denote $A\mathbf{v}_k$ by $\bar{\mathbf{v}}_k$, $AV_k$ by $\bar{V}_k$, and $A\mathbf{x}_k$ by $\mathbf{p}_k$. If we define

$$\bar{W}_k = [\mathbf{w}_1, \ldots, \mathbf{w}_{k-1}, \bar{\mathbf{w}}_k] = AV_k Q_k^{\mathrm{T}} = \bar{V}_k Q_k^{\mathrm{T}}$$

$$\bar{\mathbf{z}}_k = (\xi_1, \ldots, \xi_{k-1}, \bar{\xi}_k)^{\mathrm{T}} = Q_k \mathbf{y}_k \tag{4.4}$$

then equation (2.7) becomes

$$\bar{L}_k \bar{\mathbf{z}}_k = \beta_1 \mathbf{e}_1, \tag{4.5}$$

and we have

$$\mathbf{p}_k = A\mathbf{x}_k = \bar{W}_k \bar{\mathbf{z}}_k. \tag{4.6}$$

So, $\bar{\mathbf{v}}_i$ and $\mathbf{w}_i$ can be formed, used, and discarded one by one. It is wasteful to update $\mathbf{p}_k$ fully each step in formula (4.6), while if $\bar{L}_k$ is singular then $\bar{\mathbf{z}}_k$ is undefined. For this reason, we define $L_k$ which denotes $\bar{L}_k$ with $\bar{\gamma}_k$ replaced by $\gamma_k [= (\bar{\gamma}_k^2 + \beta_{k+1}^2)^{1/2}]$. We also define $\mathbf{z}_k = (\xi_1, \ldots, \xi_k)^{\mathrm{T}}$ and $W_k = [\mathbf{w}_1, \ldots, \mathbf{w}_k]$, where $\mathbf{z}_k$ is found from

$$L_k \mathbf{z}_k = \beta_1 \mathbf{e}_1. \tag{4.7}$$

Because $\gamma_k = (\bar{\gamma}_k^2 + \beta_{k+1}^2)^{1/2}$ and $\beta_{k+1} \neq 0$, $\mathbf{z}_k$ is always defined. If $\beta_{k+1} = 0$, the Lanczos iteration will terminate with $\mathbf{x}_k = \mathbf{x}$ (see Ref. [5, p. 623]). Rather than updating $\mathbf{p}_k$, we update

$$\mathbf{p}_k^L = \mathbf{p}_{k-1}^L + \xi_k \mathbf{w}_k. \tag{4.8}$$

From the above, it can be shown that

$$\mathbf{p}_k = \mathbf{p}_{k-1}^L + \bar{\xi}_k \bar{\mathbf{w}}_k. \tag{4.9}$$

So we are able to obtain easily $\mathbf{p}_k$ or the residual $\| \mathbf{p}_k - \mathbf{b} \|$ if it is needed.

In order to update $A\mathbf{x}_k$, i.e. $\mathbf{p}_k$, the computation cost is usually much less than that for the matrix–vector product $A\mathbf{v}_k$. We have known that $A\mathbf{x}_k$ is required to compute $\| A\mathbf{x}_k - \mathbf{b} \|$ only when the level of orthogonality reaches $\sqrt{\epsilon}$. But for subsequent steps, there is the possibility of $\bar{\gamma}_k = 0$. In such a case, $\bar{L}_k$ is singular, so is $T_k$; and $A\mathbf{x}_k$ will be undefined. Fortunately, at this step $\mathbf{x}_k$ is not a good solution to $A\mathbf{x} = \mathbf{b}$, because $A$ is not singular. Therefore in practical computation, if $\bar{\gamma}_k = 0$ we need not update $\mathbf{p}_k$ to compute the residual $\| \mathbf{p}_k - \mathbf{b} \|$. From numerical experience, the number of computations of $\| \mathbf{p}_k - \mathbf{b} \|$ is generally much less than the total number of Lanczos steps. This has been demonstrated by Example 4.1.

Finally, let us summarize our new algorithm for ill-conditioned linear equations as below. For convenience, this algorithm is referred to as LANILE.

(1) Initialization.
(2) Loop: for $k = 1, 2, \ldots,$
  (2.1) Take a simple Lanczos step.
  (2.2) Update $\omega_{ik+1}$.
  (2.3) If $\omega_{\max} \geqslant \sqrt{\epsilon}$, perform partial reorthogonalization
  (2.4) If step $m$ has not arrived, compute $\| \mathbf{r}_k \|$ by formula (2.8).
  (2.5) If step $m$ has arrived and $\bar{\gamma}_k \neq 0$, then:
    (a) update $\mathbf{p}_k$;
    (b) $\| \mathbf{r}_k \| \leftarrow \| \mathbf{p}_k - \mathbf{b} \|$.
  (2.6) If $\| \mathbf{r}_k \| \leqslant$ tolerance, end the loop and go to (3)
  (2.7) Update $\mathbf{p}_k^L$.
(3) Backsubstitute $\mathbf{y}_k = Q_k^{\mathrm{T}} \bar{L}_k^{-1} \beta_1 \mathbf{e}_1$.
(4) Assemble $\mathbf{x}_k = V_k \mathbf{y}_k$.

*Note.* Step $m$ is the step at which $\omega_{\max} \geqslant \sqrt{\epsilon}$ for the first time.

## 5. SOME MORE DETAILS

In Ref. [8], to increase the solution accuracy further, $T_k \mathbf{y}_k = \beta_1 \mathbf{e}_1$ was modified to the form

$$T_k' \mathbf{y}_k' = \beta_1 \mathbf{e}_1, \tag{5.1}$$

where $T_k'$ includes the effect of reorthogonalizations, and $T_k'$ is a complicated matrix instead of a tridiagonal one. We have tested this on some numerical examples. We found that $\mathbf{y}_k'$ from equation (5.1) gave little better results than the original $\mathbf{y}_k$. This is because reorthogonalizations of the Lanczos vectors have little effect on the basic equation (2.3), as pointed out in Section 3. In addition, this measure destroyed the tridiagonal form of $T_k$. So in the present algorithm LANILE, we do not employ equation (5.1).

Secondly, in Ref. [13] Simon used scaled QR factorization to solve equation (2.7). We also tested this on some ill-conditioned problems. We found that using this method the solution accuracy was increased only a little in most cases. Here is an example: $A$ is a 10 by 10 matrix, $A = \text{diag}(10^{-4}, 2, 3, 4, \ldots, 10)$; $\mathbf{b} = (1, 1, \ldots, 1)^T$. At step 10 the true residual norms by QR and scaled QR factorization are, respectively, $5.386 \times 10^{-3}$ and $5.037 \times 10^{-3}$. For simplicity, we did not use the scaled QR in LANILE. The above example was solved in single precision.

Finally, in SYMMLQ $\mathbf{x}_k^L$, which in theory equals $V_k Q_k^T \mathbf{z}_k$, is employed as an alternative to $\mathbf{x}_k$ if $\mathbf{x}_k^L$ gives a smaller residual norm than $\mathbf{x}_k$. In all the practical examples we have computed, $\mathbf{x}_k$ is always better than $\mathbf{x}_k^L$. What is more, the updating of $\mathbf{x}_k^L$ requires additional computations, so $\mathbf{x}_k^L$ is not included in LANILE.

## 6. NUMERICAL EXAMPLES

The algorithm LANILE has been programmed and tested on many problems in order to obtain an impression of its numerical property. Experience shows that when the condition number of the linear system exceeds $\epsilon^{-1/3}$ the system is regarded as ill-conditioned in our present algorithm, and LANILE should be used instead of LANPRO. If users do not know the condition number of their equations in advance, LANILE can always be used. The extra computation cost is not large.

We should note that the updated value $\| \mathbf{p}_k - \mathbf{b} \|$ is still different from $\| A\mathbf{x}_k - \mathbf{b} \|$ which is directly computed in such a way that $\mathbf{x}_k$ is first formed and then the product $A\mathbf{x}_k$ is calculated. However the difference is negligible, because the solution error of the ill-conditioned tridiagonal system, which is the main cause of the failure of formula (2.8) as stated above, has been included in $\mathbf{p}_k$. This can be seen from the following examples. For convenience, the original predicted relative residual ($\| \mathbf{r}_k \| / \| \mathbf{b} \|$) using formula (2.8) is represented by $\rho_k^o$, the present predicted by $\rho_k^p$, and the

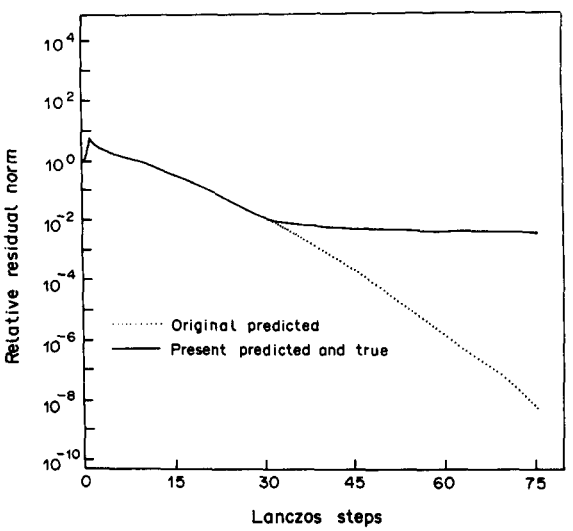


Fig. 2. Predicted and true residual for Example 6.1.

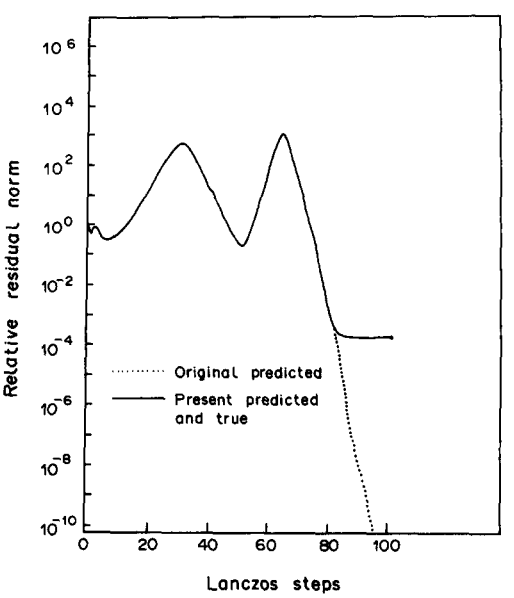


Fig. 3. Predicted and true residual for Example 6.2.

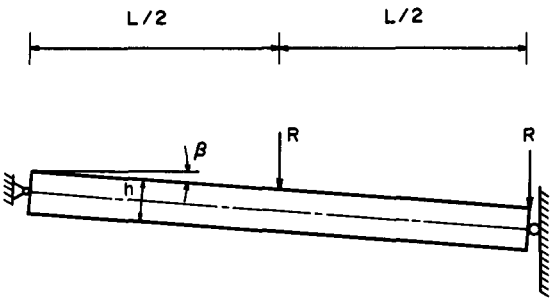


Fig. 4. Structure with ill-conditioned stiffness matrix: $L = 2000$ mm; $h = 20$ mm; $t = 4$ mm (width); $\beta = 0.01$; $E = 200$ kN/mm$^2$; $\nu = 0.3$ (poisson's ratio).

true relative residual by $\rho_k^1$. In all computations of this paper, the round-off unit $\epsilon = 2^{-24}$ for single precision; $\epsilon = 2^{-56}$ for double precision.

*Example 6.1.* This example is a 1000 by 1000 linear system. $A = \text{diag}(2^{-1}, 4^{-1}, 6^{-1}, \ldots, 2000^{-1})$, and $\mathbf{b} = (1, 1, \ldots, 1)^T$. In single precision, this problem can be considered ill-conditioned. The algorithm was stopped when $\rho_k^0 < 10^{-8} (< \epsilon)$, which occurred at step 75. The results are shown in Fig. 2. The values of $\rho_{75}^0$, $\rho_{75}^p$, and $\rho_{75}^1$ are, respectively, $0.4243 \times 10^{-8}$, $0.6824 \times 10^{-2}$ and $0.6816 \times 10^{-2}$. We can see from Fig. 2 that the curves of $\rho_k^p$ and $\rho_k^1$ are identical.

*Example 6.2.* This example is a 100 by 100 indefinite ill-conditioned system. $A = \text{diag}(10^{-7}, -100, 6, 8, 10, \ldots, 198, 10^{-6})$, $\mathbf{b} = (1, 1, \ldots, 1)^T$. In double precision computation, at the 100th step, $\rho_k^0$, $\rho_k^p$, and $\rho_k^1$ are, respectively, $0.3206 \times 10^{-33}$, $0.2255 \times 10^{-3}$ and $0.2255 \times 10^{-3}$. The results are shown in Fig. 3.

*Example 6.3.* This problem arises from the finite element approximation to the structure as shown in Fig. 4. Using 220 four-node plane stress elements, we discretize the structure to a total of 663 degrees of freedom. It is obvious that the initial stiffness matrix is ill-conditioned. We employ the LANILE to solve the ill-conditioned linearized system of equations arising at the first iterate of the Newton–Raphson iteration scheme for analyzing the nonlinear structure. The first load increment is 0.4 kN. In this example, most elements of the right-hand side **b** are zero, which is different from the previous **b**s.

In double precision, the algorithm was terminated when $\rho_k^0 < 10^{-18}$ $(< \epsilon)$, which occurred at step 570. The values of $\rho_{570}^0$, $\rho_{570}^p$, and $\rho_{570}^1$ are, respectively, $0.5242 \times 10^{-18}$, $0.5135 \times 10^{-2}$ and $0.5135 \times 10^{-2}$.

## REFERENCES

1. C. Lanczos, Solution of systems of linear equations by minimized iterations. *J. Res. natn. Bur. Stand.* **49**, 33–53 (1952).
2. C. Paige, Computational variants of the Lanczos method for the eigenproblem. *J. Inst. Math. Appl.* **10**, 373–381 (1972).
3. W. Kahan and B. N. Parlett, How far can you go with the Lanczos algorithm? *Sparse Matrix Computations* (Eds J. Bunch and D. Rose). Academic Press, New York (1976).
4. J. Cullum and R. Willoughby, Computing eigenvectors and eigenvalues of large sparse symmetric matrices using Lanczos tridiagonalization. *Numerical Analysis Proceedings, Dundee 1979* (Ed. G. A. Watson). Springer, Berlin (1980).
5. C. Paige and M. Saunders, Solution of sparse indefinite systems of linear equations. *SIAM Jl numer. Analysis* **12**, 617–629 (1975).
6. B. N. Parlett and D. Scott, The Lanczos algorithm with selective orthogonalization. *Math. Comput.* **33**, 217–238 (1979).
7. B. N. Parlett, *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, N.J. (1980).
8. B. N. Parlett, A new look at the Lanczos algorithm for solving symmetric systems of linear equations. *Linear Algebra Applic.* **29**, 323–346 (1980).
9. H. D. Simon, The Lanczos algorithm with partial reorthogonalization. *Math. Comput.* **42**, 115–142 (1984).
10. B. Nour-Omid, B. N. Parlett and R. L. Taylor, Lanczos versus subspace iteration for solution of eigenvalue problems. *Int. J. numer. Meth. Engng* **19**, 859–871 (1983).
11. B. Nour-Omid, B. N. Parlett and R. L. Taylor, A Newton–Lanczos method for solution of nonlinear finite element equations. *Comput. Struct.* **16**, 241–252 (1983).
12. A. L. G. A. Coutinho, J. L. D. Alves, E. C. P. Lima and N. F. F. Ebecken, On the application of an element-by-element Lanczos solver to large offshore structural engineering problems. *Comput. Struct.* **27**, 27–37 (1987).
13. H. D. Simon, The Lanczos algorithm for solving symmetric linear systems. Ph.D. Thesis, University of California, Berkeley (1982).